Tugas Kecil 3 IF 2211 Strategi Algortima

Aplikasi Algoritma *Branch and Bound* untuk Membuat Suatu Program Solver dari Permainan Puzzle 15

Dibuat oleh :

Nama : Timothy Stanley Setiawan

NIM : 13520028

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia

Daftar Isi

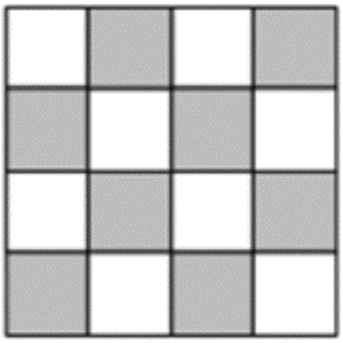
BAB 1: Cara Kerja Program Branch and Bound yang Dibuat Dalam Menyelesaikan Persoalan

Pada tucil 3 mata kuliah strategi algoritma, penulis memanfaatkan algoritma *Branch and Bound* dalam pembentukan program *solver* dari permainan Puzzle 15. Berikut ini penjelasan algoritam yang penulis gunakan.

1	2	3	4
5	6		8
9	10	7	11
13	14	15	12

Gambar 1 Contoh Puzzle

1. *Prerequisite satu*, untuk mencari tahu apakah puzzle yang dimasukkan dapat diselesaikan atau tidak. Perlu dicari nilai dari $\sum_1^{16} Kurang(i) + X$ bernilai genap atau ganjil.
2. *Prerequisite kedua*, untuk mencari nilai dari masing-masing fungsi Kurang(i) nilai fungsi Kurang(i) didapatkan dengan cara menghitung banyak ubin bernomor j sehingga $j < i$, akan tetapi $Posisi(j) > Posisi(i)$. Di sisi lain, untuk mencari nilai X lihat gambar di bawah ini apabila posisi kotak kosong berada pada daerah yang diarsir, X bernilai 1, sebaliknya bernilai 0



Gambar 2 Gambar puzzle yang diarsir untuk menentukan nilai X

i	Kurang (i)
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	1
9	1
10	1
11	0
12	0
13	1
14	1
15	1
16	9

Gambar 3 Nilai fungsi Kurang(i) untuk masing-masing i
dari puzzle pada gambar 1

3. *Prerequisite ketiga*, tentukan apakah puzzle dapat diselesaikan atau tidak? Jika tidak maka keluarkan pesan "tidak bisa menyelesaikan puzzle" jika iya maka lanjutkan ke langkah Pertama.

***Penting!!!**:*

Algoritma branch and bound yang digunakan dalam program solver Puzzle 15 ini memanfaatkan tipe data priorityqueue yang menyimpan queue berupa node yang memiliki informasi parent node, kondisi puzzle, kedalaman simpul, list_move yang sudah diambil, dan cost untuk menyelesaikan puzzle pada simpul yang bangkitkan. Priorityqueue ini akan menempatkan node dengan kalkulasi cost terendah pada queue paling awal. Untuk kalkulasi cost, didapatkan dari fungsi $\hat{c}(i) = \hat{f}(i) + \hat{g}(i)$. $\hat{f}(i)$ *adalah ongkos untuk mencapai simpul i dari akar, yaitu panjang lintasan (depth) yang dilalui dari akar ke simpul i.* $\hat{g}(i)$ *adalah ongkos untuk mencapai simpul tujuan dari simpul i, yaitu taksiran lintasan terpendek dari simpul i ke simpul posisi dengan nilai taksiran ini didapatkan dengan mencari jumlah ubin tidak kosong yang letaknya tidak sesuai dengan posisi pada susunan final. Selain itu, type data hash table juga digunakan untuk menyimpan simpul yang sudah pernah dibangkitkan sehingga simpul yang sudah pernah dibangkitkan tidak akan dibangkitkan kembali. Alasan memilih hash table untuk menyimpan simpul-simpul yang sudah dibangkitkan ketimbang list adalah ketika menggunakan hash table proses pengecekan apakah*

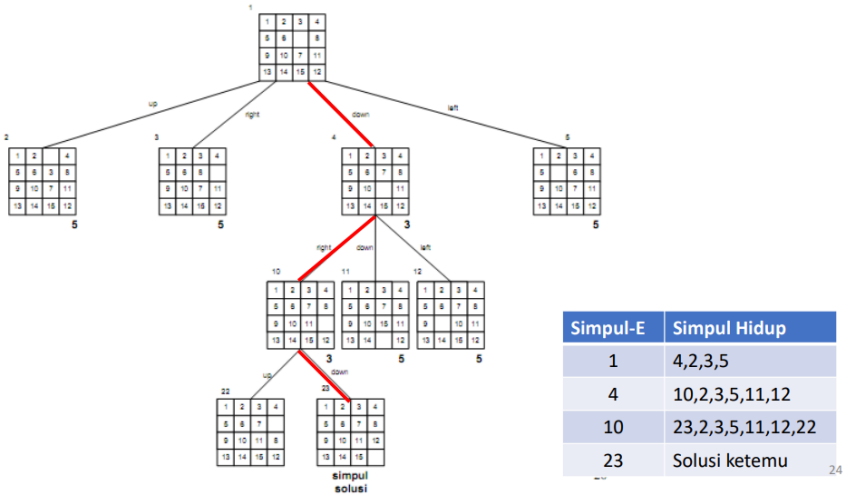
simpul sudah pernah dibangkitkan atau belum lebih cepta ketimbang list, walau harus mengorbankan memori. Terakhir, ada type data linked list (node) untuk mempermudah tracking *dari simpul goal ke akar (ketika mencapai simpul final)*

4. Pertama, masukan simpul akar dengan semua informasi yang diperlukan ke dalam priorityqueue kosong yang sudah dibuat.
5. Kedua, lakukan dequeue untuk mengambil simpul pertama dalam queue kemudian dicek apakah simpul tersebut adalah simpul goal atau tidak jika iya lompat ke langkah kelima jika tidak lanjutkan ke langkah ketiga. Simpul inilah yang kemudian akan di-*ekspan* (jika bukan simpul goal)
6. Ketiga, cek semua kemungkian pergeseran yang aman untuk dilakukan kotak kosong pada simpul saat itu. Pergeseran yang aman adalah pergeseran yang membuat kotak kosong tidak keluar dari puzzle. Misalnya untuk dibawah ini pergerseran yang aman adalah bawah, kiri, atas karena dari semua kemungkinan pergeseran tadi masih membuat kotak kosong di dalam puzzle. Sedangkan untuk pergeseran ke kanan tidak aman karena kotak kosong akan keluar dari puzzle (keluar batas)

1	2	3	4
5	6	8	
9	10	7	11
13	14	15	12

Gambar 4 Kondisi puzzle pada simpul i, pada simpul ini arah yang aman untuk melakukan pergeseran adalah bawah, kiri, atas

7. Keempat, bangkitkan semua simpul yang mungkin, yaitu simpul dengan arah pergeseran yang aman dan juga simpul tersebut belum pernah dibangkitkan sebelumnya. Simpul yang sesuai syarat pembangkitan akan dibangkitkan berserta informasi-informasi yang diperlukannya. Kemudian kembali ke langkah kedua.
8. Kelima, ketika simpul goal ditemukan, proses pencarian simpul (langkah kedua sampai keempat) dihentikan dan keluarkan output berupa lintsan simpul yang dilalui dari simpul akar ke simpul goal berserta dengan infomasi waktu pencarian, kedalaman simpul, dan jumlah simpul yang dibangkitkan.



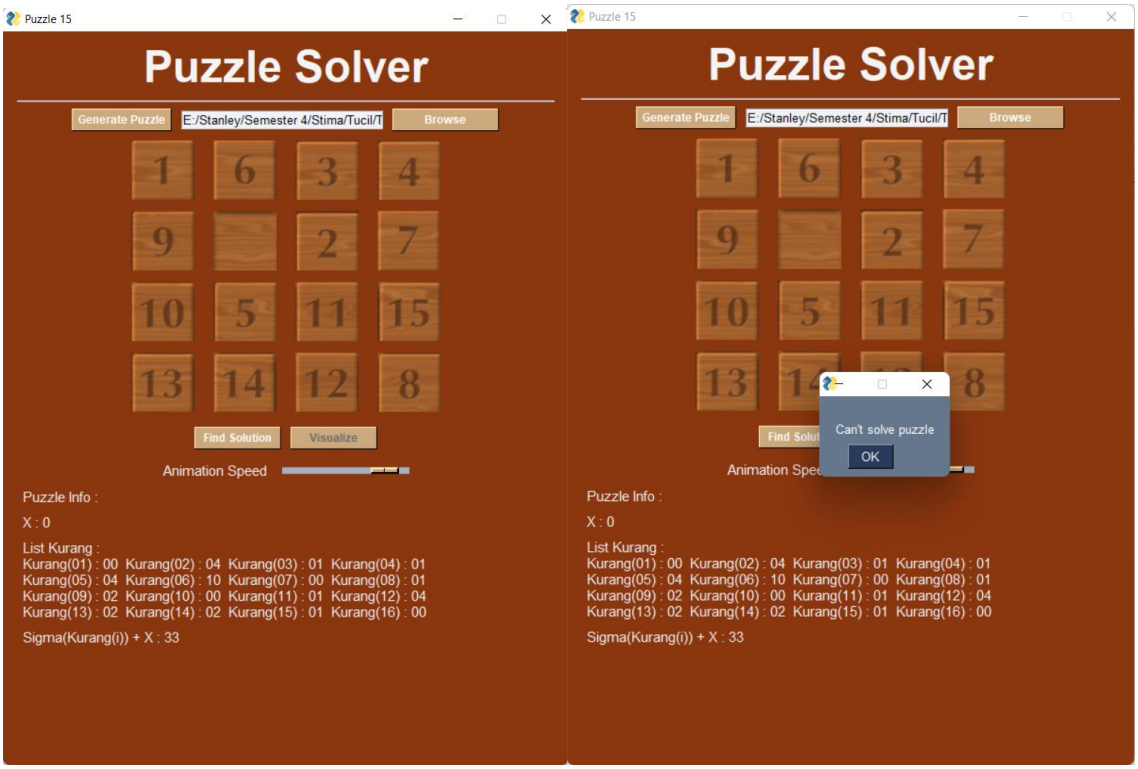
Simpul-E	Simpul Hidup
1	4,2,3,5
4	10,2,3,5,11,12
10	23,2,3,5,11,12,22
23	Solusi ketemu

Gambar 5 contoh dari pohon ruang status yang terbentuk dari pencarian solusi

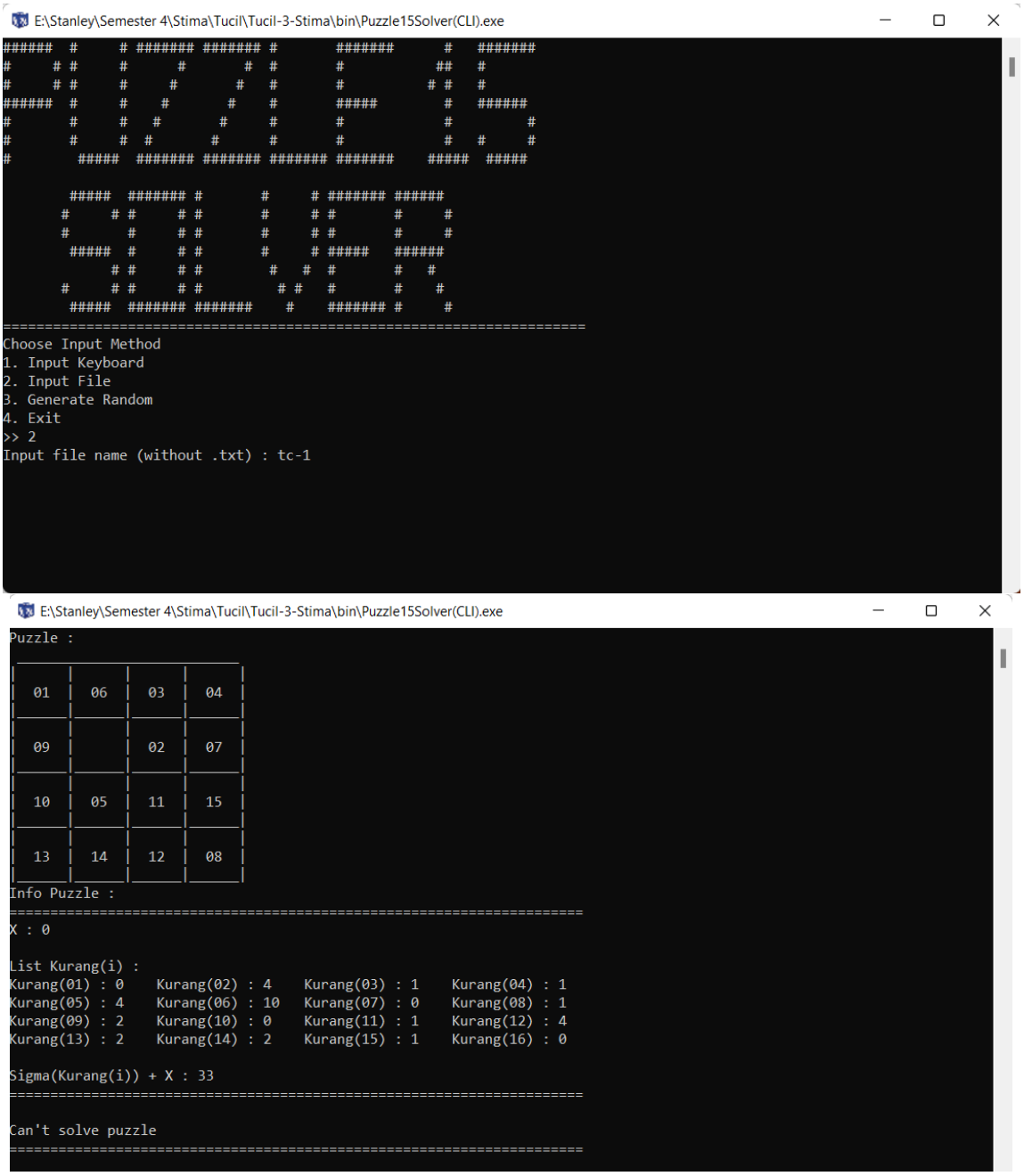
Catatan:
Karena menggunakan priorityqueue dengan prioritas queue adalah cost untuk menyelesaikan puzzle, dijamin setiap kali pemilihan simpul (dequeue) dijamin merupakan simpul dengan cost terendah. Selain itu, program ini juga membatasi proses iterasi hingga 1 juta iterasi agar tidak terjadi infiniti loop/ pemrosesan program yang terlalu lama apabila terjadi (khususnya ketika puzzle yang akan di solve terlalu kompleks).

BAB 2: *Screen Shoot Input Output Program*

2.1. tc-1.txt (Test case tidak bisa diselesaikan)

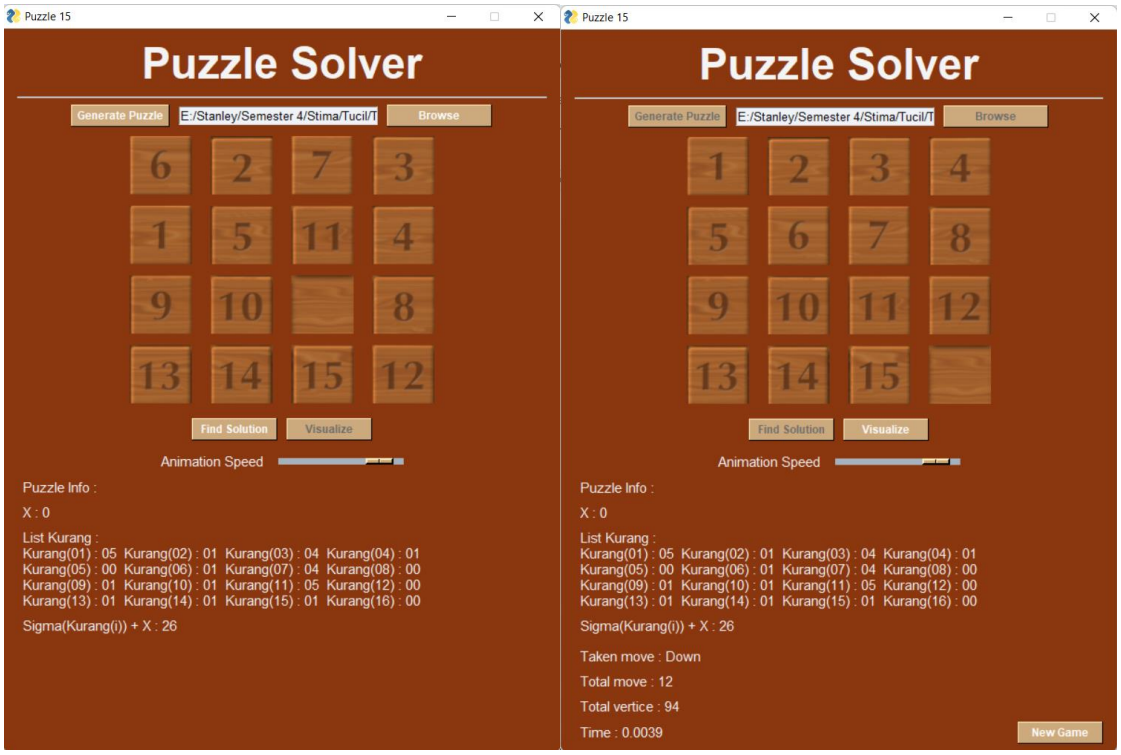


Gambar 6 screen shot program untuk tc-1 secara GUI

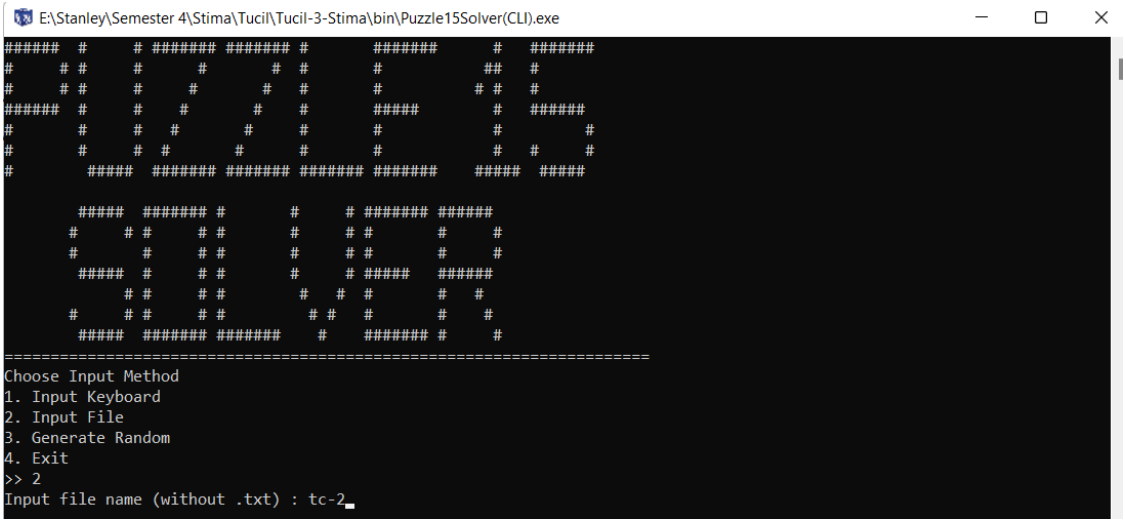


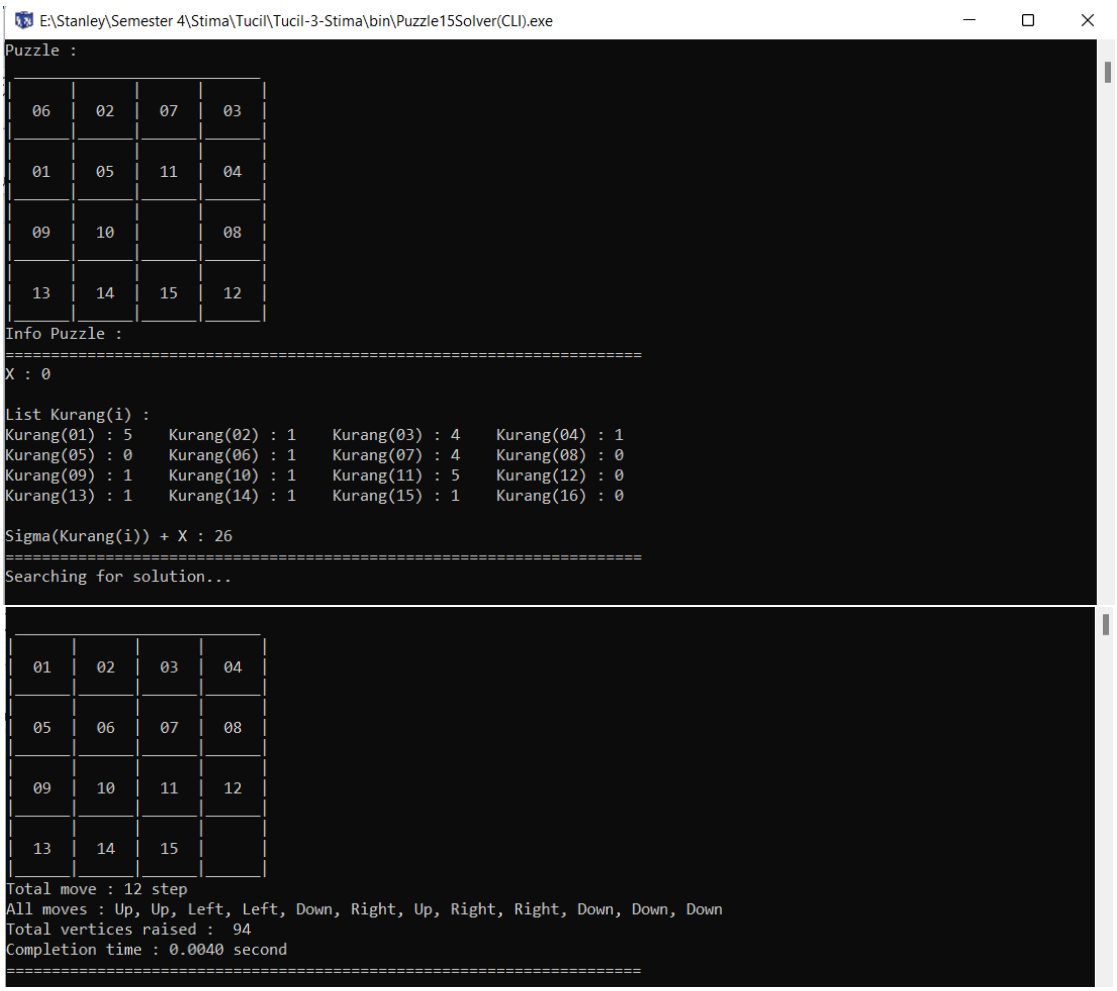
Gambar 7 screen shot program untuk tc-1 secara CLI

2.2. tc-2.txt (Test case bisa diselesaikan, level mudah)



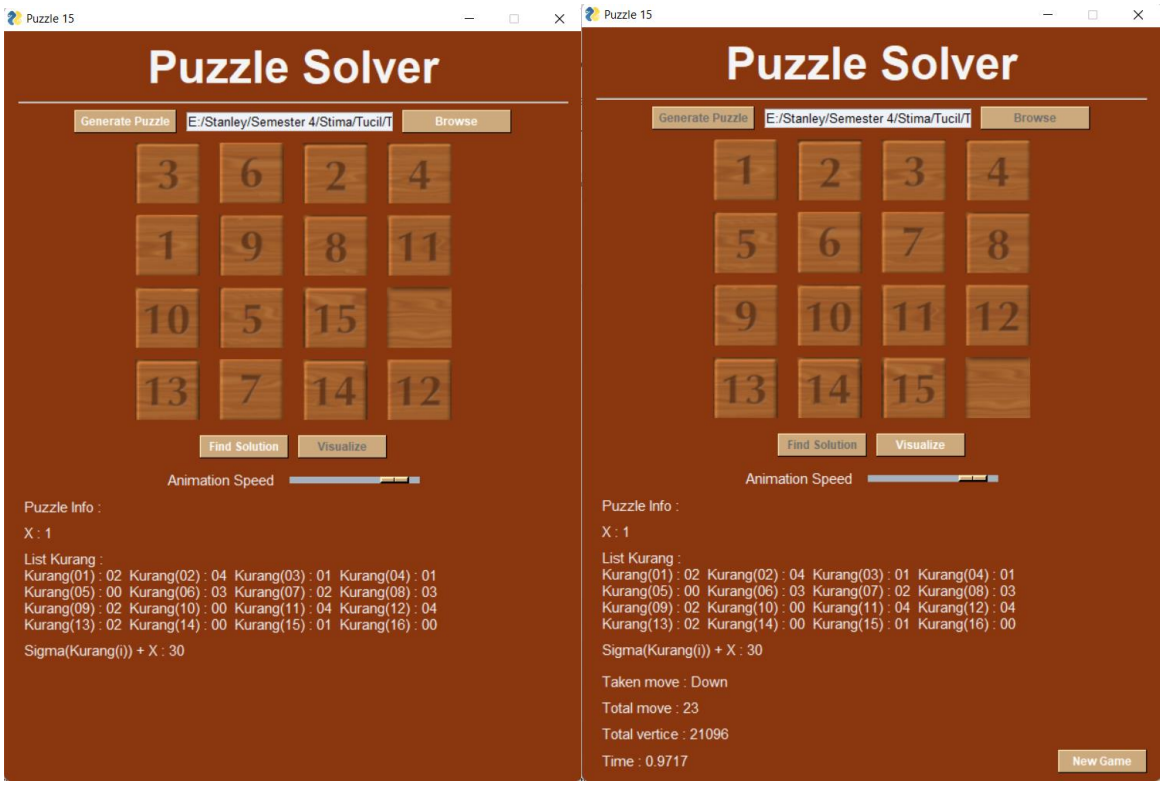
Gambar 8 screen shot program untuk tc-2 secara GUI



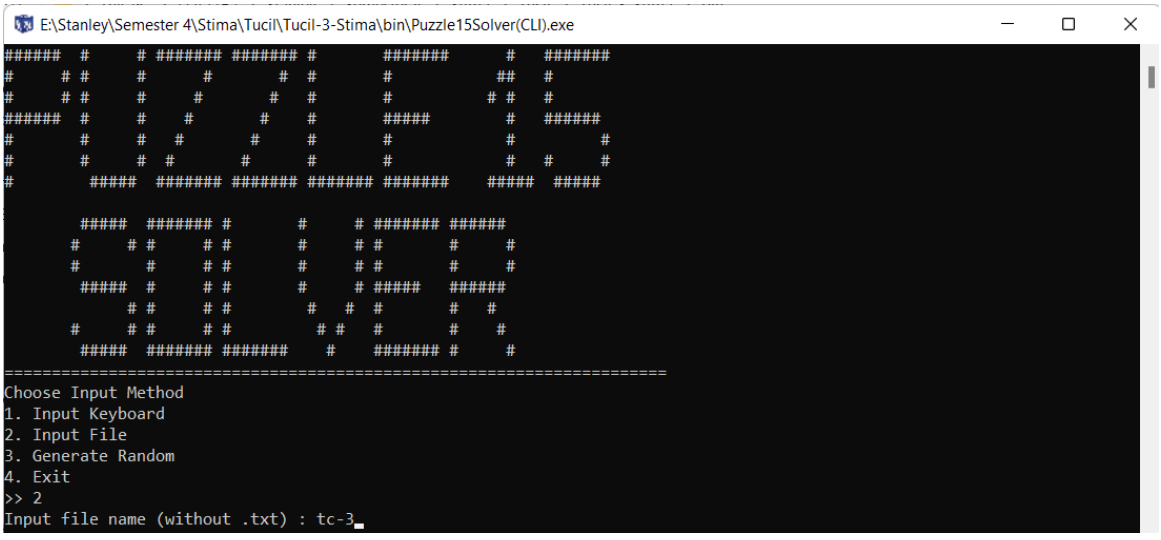


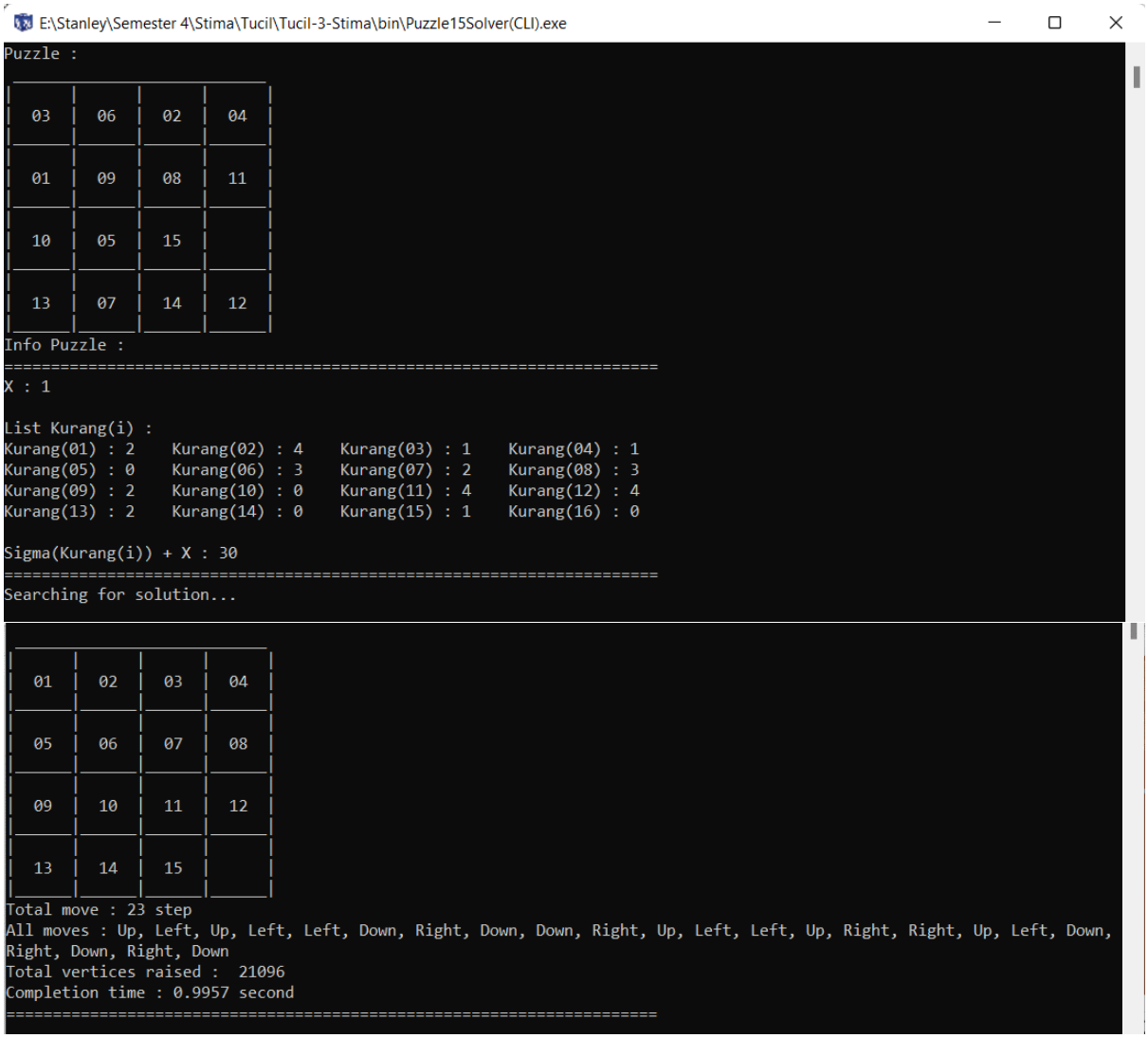
Gambar 9 screen shot program untuk tc-2 secara CLI

2.3. tc-3.txt (Test case bisa diselesaikan sedang)



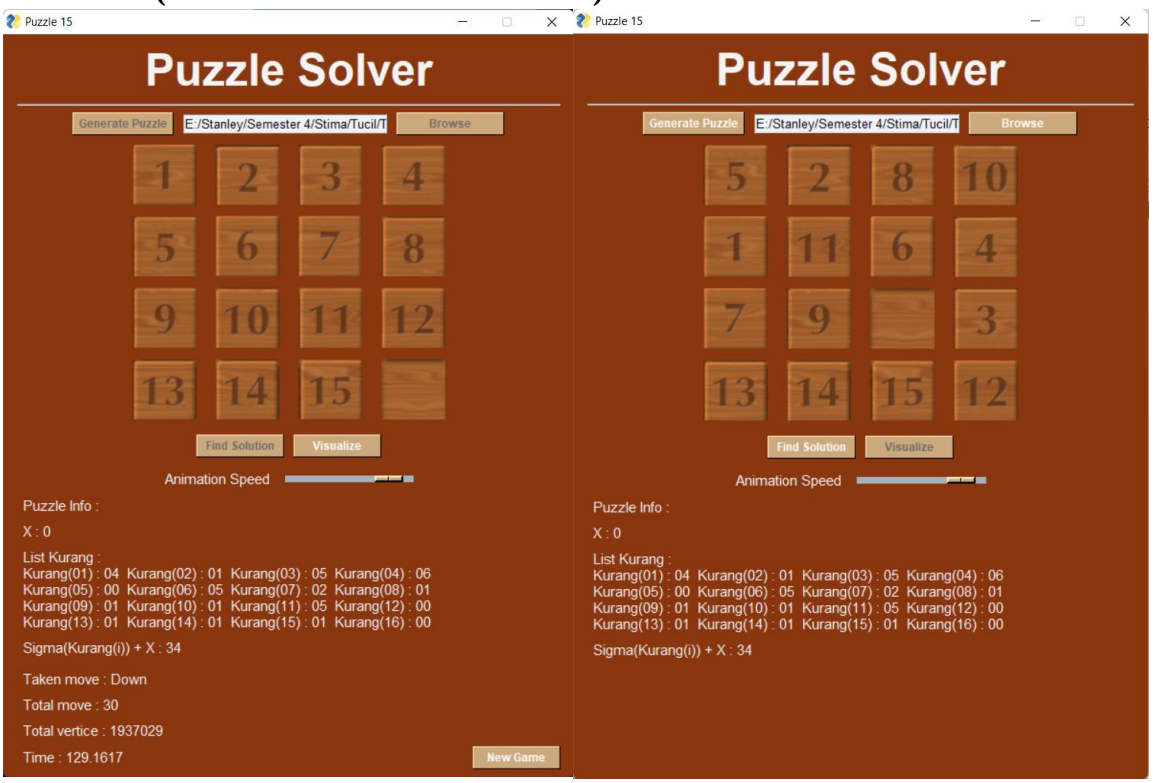
Gambar 10 screen shot program untuk tc-3 secara GUI



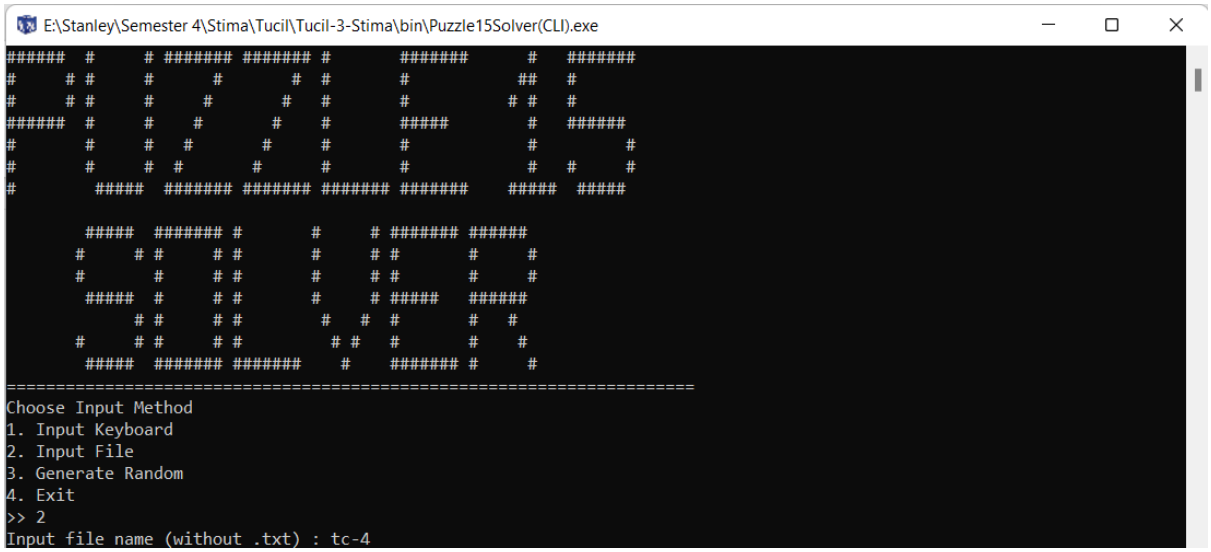


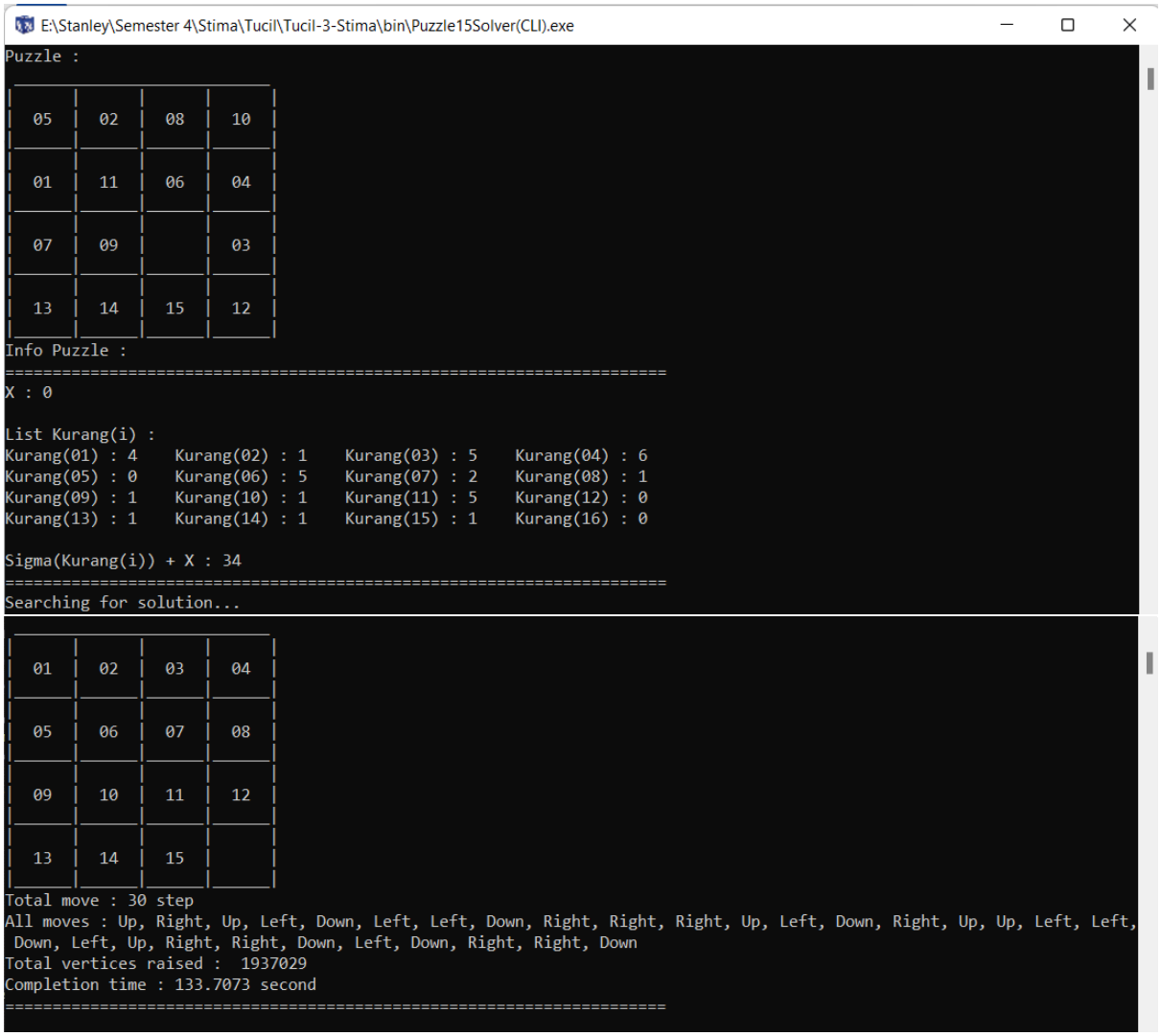
Gambar 11 screen shot program untuk tc-3 secara CLI

2.4. tc-4.txt (Test case bisa diselesaikan sulit)



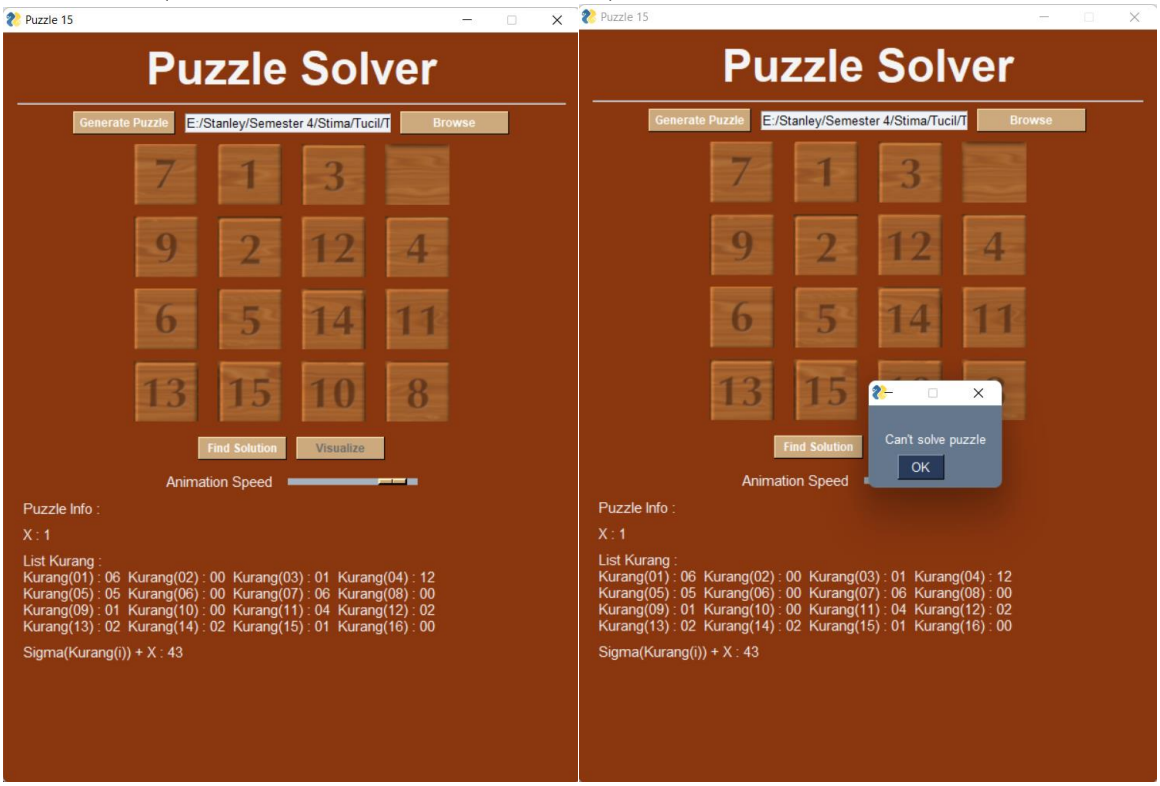
Gambar 12 screen shot program untuk tc-4 secara GUI



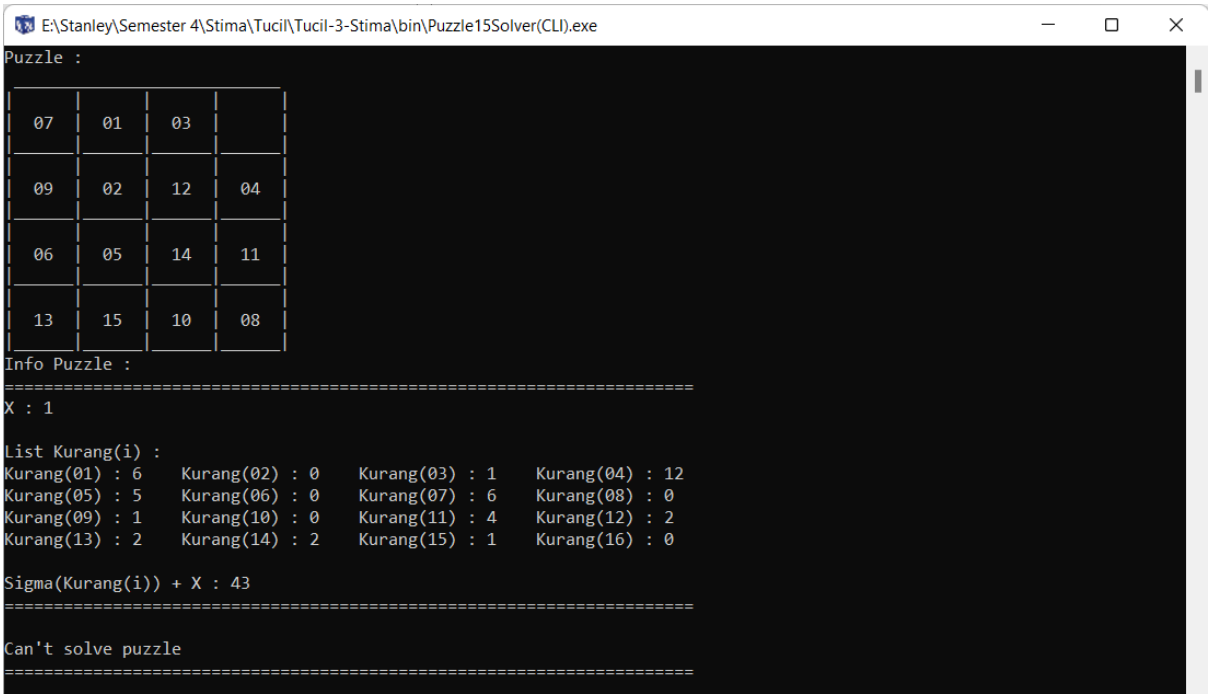


Gambar 13 screen shot program untuk tc-1 secara CLI

2.5. tc-5.txt (Test case tidak bisa diselesaikan)

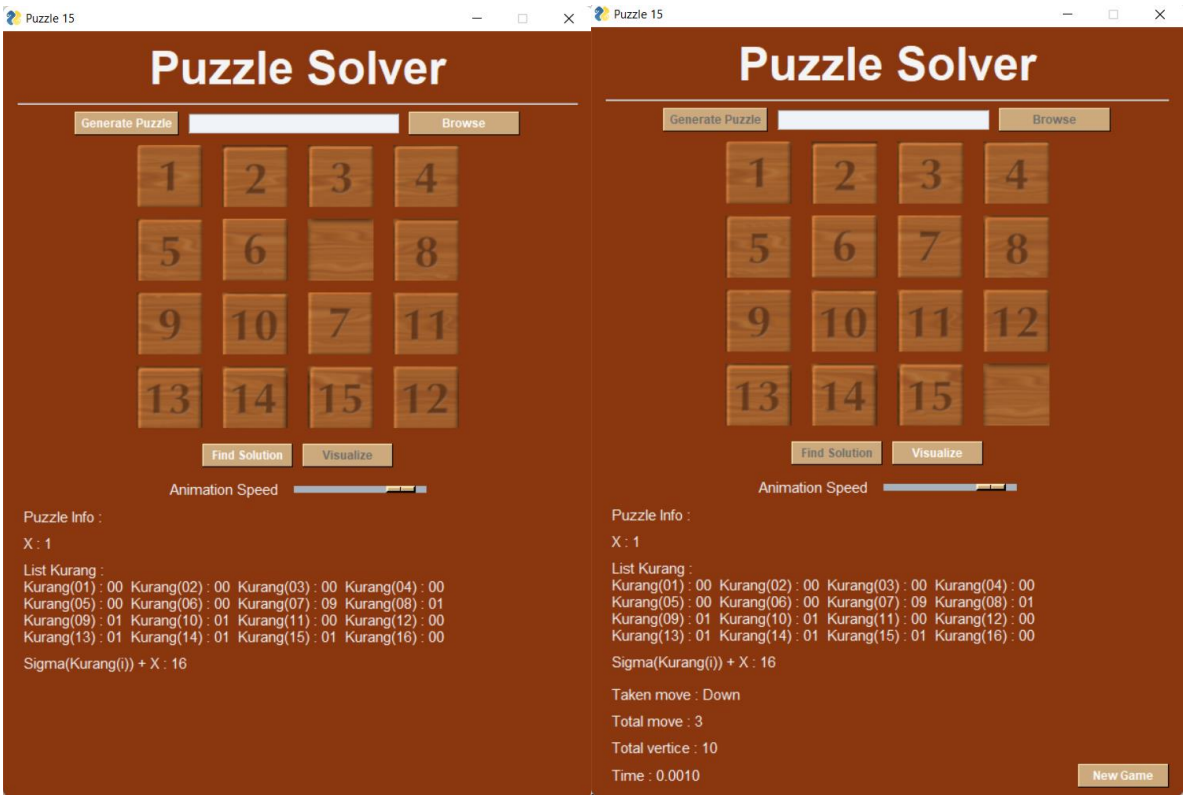


Gambar 14 screen shot program untuk tc-5 secara GUI

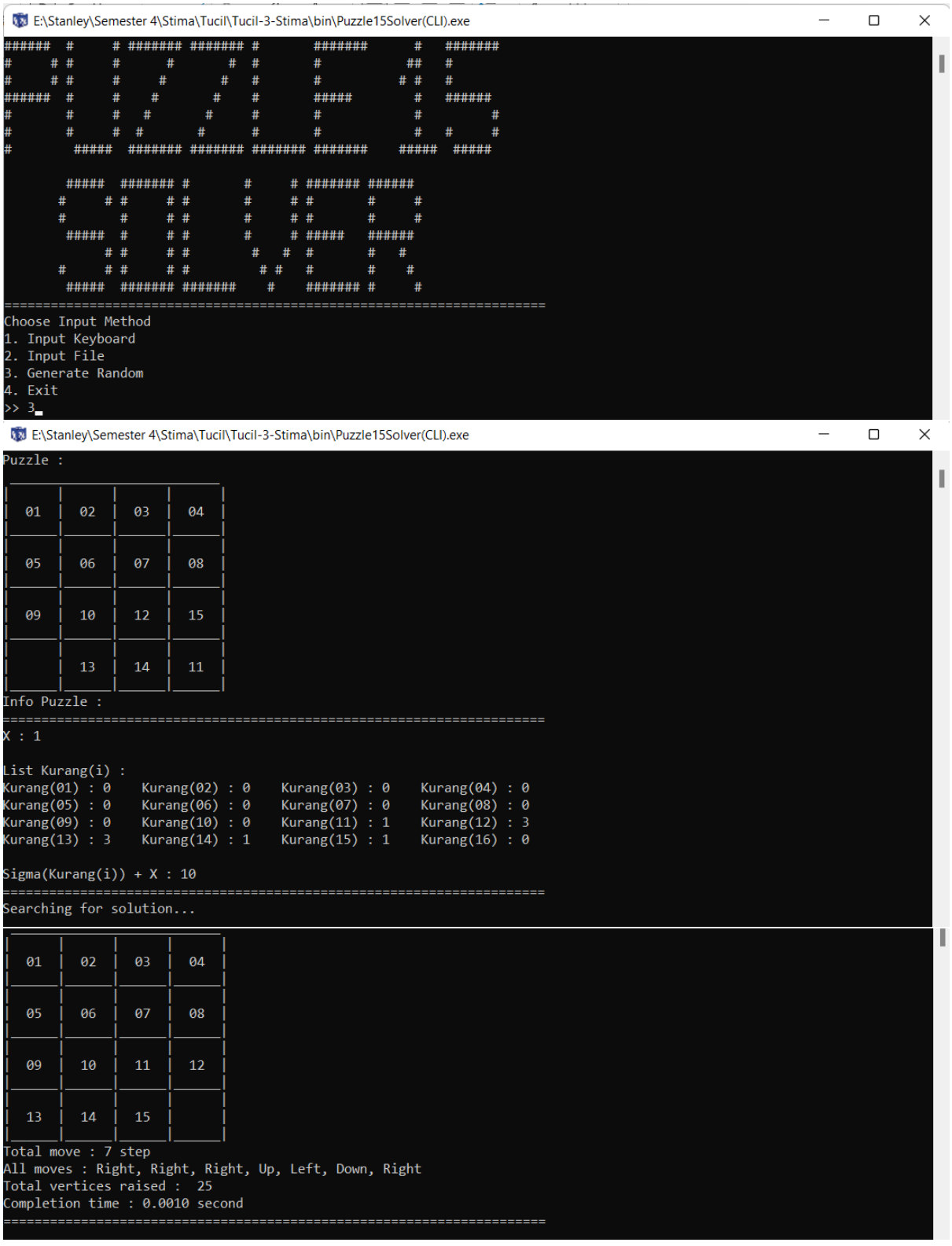


Gambar 15 screen shot program untuk tc-5 secara CLII

2.6. generate random puzzle

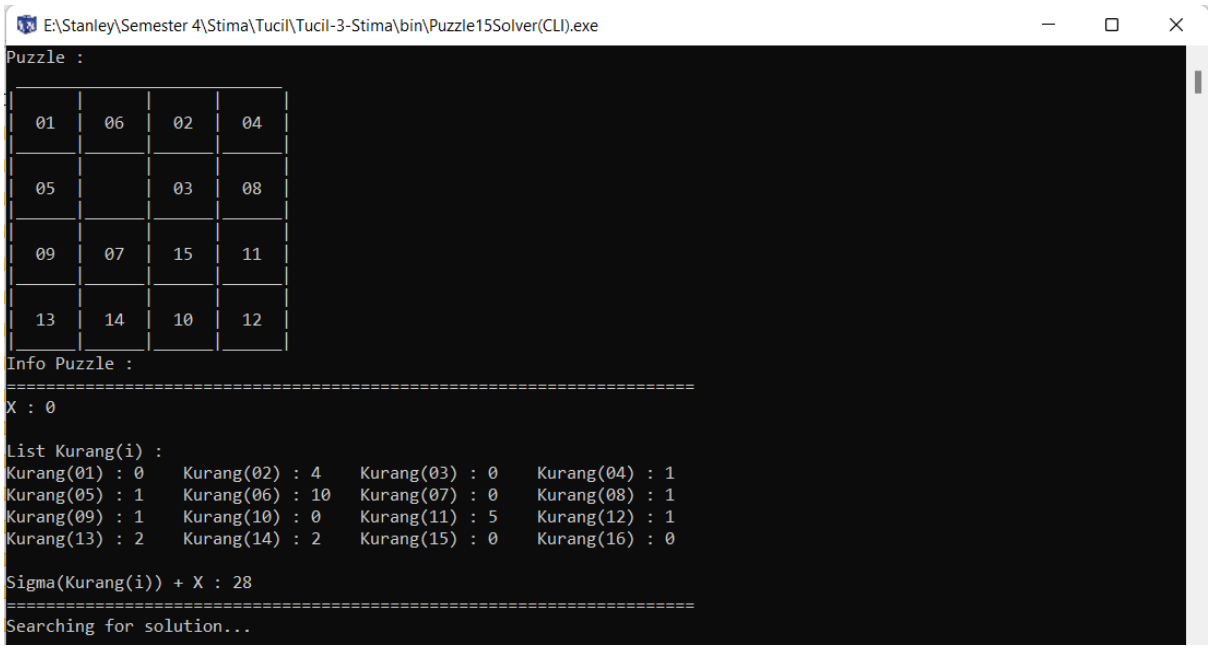


Gambar 16 screen shot program untuk generate puzzle random secara GUI



Gambar 17 screen shot program untuk generate puzzle random secara CLI

2.7. puzzle input user (hanya bisa di cli)



Gambar 18 screen shot program untuk generate puzzle random secara CLI

BAB 3: *Source Code Program* dalam bahasa Python

3.1. lib.py

```
from copy import deepcopy
from heapq import heappush, heappop
from random import randint

# class priority queue
class PriorityQueue:
    def __init__(self):
        self.elements = [] # inisialisasi awal priority queue

    def empty(self):
        return len(self.elements) == 0 # apakah priority queue kosong

    def enqueue(self, node):
        heappush(self.elements, node) # menambahkan node ke priority queue

    def dequeue(self):
        return heappop(self.elements) # mengambil node paling awal dari priority queue

# class node
class Node:
    count = 0 # inisialisasi awal counter
    step = 1 # inisialisasi awal step untuk keperluan print move
    def __init__(self, parent = None, puzzle = None, depth = None, cost = None,
list_move = None):
        self.id = "Node " + str(Node.count) # inisialisasi id node
        self.parent = parent # inisialisasi parent node (untuk root parent = None)
        self.puzzle = puzzle # inisialisasi puzzle node
        self.depth = depth # inisialisasi depth node
        self.cost = cost # inisialisasi cost node
        self.list_move = list_move # inisialisasi list move node
        Node.count += 1 # menambah counter untuk id
        Node.step = 1 # inisialisasi awal step

    # operator overloading kurang dari untuk keperluan priority queue
    def __lt__(self, other):
        return self.cost < other.cost

    # melakukan print info node digunakan ketika debugging
    def printNode(self):
        print("------------------")
        print("ID : ", self.id)
        if (self.parent == None):
            print("ParentID : None")
```

```
        else:
            print("ParentID : ", self.parent.id)
        print("Puzzle : ")
        self.puzzle.printPuzzle()
        print("Depth : ", self.depth)
        print("Cost : ", self.cost)
        print("------------------")

    # print semua node lintasan dari akar ke simpul goal
    def printall(self):
        if (self.parent == None):
            self.step = 1
            self.puzzle.printPuzzle()
            return
        self.parent.printall()
        print("Step : ", Node.step, "\nMove : ", self.list_move[Node.step-
1].capitalize())
        Node.step += 1
        print()
        print("         ⬇             ")
        print()
        self.puzzle.printPuzzle()

    # menyimpan semua node lintasan dari akar ke simpul goal ke dalam list untuk
keperluan GUI
    def node_to_list(self):
        list_puzzle = []
        p = self
        while (p != None):
            list_puzzle.append(p.puzzle)
            p = p.parent
        return list_puzzle

# class puzzle
class Puzzle:
    row = 4 # nilai default row puzzle
    col = 4 # nilai default col puzzle
    # nilai default goal
    #  1  2  3  4
    #  5  6  7  8
    #  9 10 11 12
    # 13 14 15 16
    default_goal = [1,2,3,4,
                    5,6,7,8,
                    9,10,11,12,
                    13,14,15,16]
    def __init__(self, buffer = []):
```

```
        self.buffer = buffer # inisialisasi awal buffer puzzle
        self.blank = None # inisialisasi awal posisi kosong
        self.X = None # inisialisasi awal posisi X
        self.list_kurang = [] # inisialisasi awal list nilai fungsi kurang(i)
        self.sigma_kurang = None # inisialisasi jumlah semua nilai kurang(i) puzzle

    # mendapatkan buffer puzzle
    def get(self, i):
        return self.buffer[i]

    # mengisi nilai X
    def setX(self, i, j):
        # mengisi nilai posisi kosong
        self.blank = i *Puzzle.col + j
        # apabila i-j-1 adalah genap maka nilai X adalah 1 (bagian yang diarsir)
        if ((i-j-1)%2 == 0):
            self.X = 1
        # apabila i-j-1 adalah ganjil maka nilai X adalah 0 (bagian yang tidak diarsir)
        else:
            self.X = 0

    # mengubah nilai posisi kosong untuk swap
    def setBlank(self, new_pos):
        current_pos = self.blank
        current_elmt = self.buffer[new_pos]
        self.blank = new_pos
        self.buffer[current_pos] = current_elmt
        self.buffer[new_pos] = 16

    # melakukan read file tc
    def read(self, filename, gui = False):
        # jika bukan dibaca melalui GUI, default path adalah test/[filename].txt
        if (gui == False):
            path = '../test/' + filename + '.txt'
        else:
        # jika dibaca melalui GUI
            path = filename
        # check apakah file ada
        try:
            # proses pembaca file
            with open(path, 'r') as f:
                lines = f.readlines()
                for i, line in enumerate(lines):
                    elements = line.split()
                    for j, element in enumerate(elements):
                        self.buffer.append(int(element))
                        if (int(element) == 16):
```

```
                            # mengisi nilai X
                            self.setX(i,j)
                # mengisi nilai sigma kurang(i) + X
                self.sigma_kurang = self.kurang()
        except:
            print('Error: Cannot open file')
            raise Exception("File not found")

    # generate puzzle random
    def generate_puzzle(self, moves = 10):
        # deepcopy default goal ke buffer
        self.buffer = deepcopy(Puzzle.default_goal)
        self.blank = 15

        # menyimpan move yang sudah dilakukan
        list_move = []
        list_move.append(self.blank)
        temp_puzzle = self # menyimpan address self

        # melakukan generate puzzle
        while moves > 0:
            swap_list = self.safe_swap()
            ran_num = randint(0,3)

            # simpan puzzle pada saat ini
            current_puzzle = deepcopy(self)

            # melakukan swap random dari list swap yang aman
            self.swap(swap_list[ran_num])

            i = 0
            # mengecek apakah swap random sudah pernah dilakukan atau belum
            while self.blank in list_move or swap_list[ran_num] == "-":
                # jika sudah dilakukan maka random swap kembali
                if (self.blank in list_move):
                    self = deepcopy(current_puzzle)
                    self.swap(swap_list[ran_num])
                # jika sudah melakukan random hingga 1000 dan selalu swap ke state yang
sama maka hentikan loop
                # mencegah infinity loop
                if (swap_list[ran_num] != "-"):
                    if (i < 1000):
                        break
                ran_num = randint(0,3)
                i += 1

            # menyimpan swap yang dilakukan
```

```
            list_move.append(self.blank)
            moves -= 1

        # jika hasil random sama dengan goal lakukan random ulang
        if (self.buffer == self.default_goal):
            self.generate_puzzle(moves)

        # mencari kolom dan baris dari posisi kosong
        i, j = int(self.blank/Puzzle.col), self.blank%Puzzle.col

        # mengisi informasi nilai buffer ke address yang sudah disimpan
        temp_puzzle.buffer = self.buffer
        # mengisi nilai X
        temp_puzzle.setX(i,j)
        # mengisi nilai sigma kurang(i) + X
        temp_puzzle.sigma_kurang = temp_puzzle.kurang()

    # mengubah puzzle menjadi string
    def puzzle_to_string(self):
        return str(self.buffer)

    # membuat goal state bisa menggunakan default goal bisa tidak
    def generate_goal(self, final = default_goal):
        self.buffer = final

    # print semua informasi puzzle
    def printInfo(self):
        print('Puzzle : ')
        # print puzzle
        self.printPuzzle()
        print("Info Puzzle :")
        print("========================================================================")
        # print nilai X
        print('X :', self.X)
        print()
        print("List Kurang(i) : ")
        # print nilai kurang(i) dengan i = 1 -16
        for i in range (Puzzle.row):
            for j in range (Puzzle.col):
                print("Kurang(%.2d) : " %((i*Puzzle.col+j)+1), end="")
                print(str(self.list_kurang[i*Puzzle.col+j]).ljust(2), end = "   ")
            print()
        print()
        # print nilai sigma kurang(i) + X
        print("Sigma(Kurang(i)) + X :", self.sigma_kurang)
        print("========================================================================")
```

```
    # print puzzle
    def printPuzzle(self):
        print(" _____________________________")
        for i in range(Puzzle.row):
            print("|      |      |      |      |")
            for j in range(Puzzle.col):
                if (j == 0):
                    print("|", end = "")
                if (self.buffer[i*Puzzle.col + j] == 16):
                    print("     ".ljust(3) + " " , end = "|")
                else:
                    print(str("  %.2d  " %self.buffer[i*Puzzle.col + j]).ljust(3),
end='|')
            print()
            print("|______|______|______|______|")

    # mencari nilai sigma kurang(i) + X dan masing-masing nilai kurang(i)
    def kurang(self):
        sum = 0
        temp = 0
        row = Puzzle.row
        col = Puzzle.col
        for i in range(row*col):
            temp = 0
            for j in range (i,row*col):
                if (self.buffer[i] > self.buffer[j]):
                    sum += 1
                    temp += 1
            self.list_kurang.append(temp)
        sum += self.X
        return sum

    # mencari swap yang aman
    def safe_swap(self):
        col = Puzzle.col
        row = Puzzle.row
        pos = self.blank
        # urutan list swap kiri, bawah, kanan, atas
        list_direct = ['-', '-', '-', '-']
        if ((pos+1)%col == 0): # swap ke kiri aman
            list_direct[2] = 'left'
        elif ((pos+1)%col == 1): # swap ke kanan aman
            list_direct[0] = 'right'
        else:   # swap ke kiri dan kanan aman
            list_direct[0] = 'right'
            list_direct[2] = 'left'
        # swap ke bawah aman
```

```
        if ((pos+1) <= row*col - col and pos >= 0):
            list_direct[3] = 'down'
        # swap ke atas aman
        if ((pos+1) > col and pos < row*col):
            list_direct[1] = 'up'
        return list_direct

    # swap puzzle
    def swap(self, direct):
        new_pos = self.blank
        match direct:
            case 'up':
                new_pos -= Puzzle.col
            case 'down':
                new_pos += Puzzle.col
            case 'left':
                new_pos -= 1
            case 'right':
                new_pos += 1
        # set posisi blank baru
        self.setBlank(new_pos)
```

3.2. bnb.py

```
from copy import deepcopy
from lib import PriorityQueue, Node, Puzzle
from time import time

# class algoritma BnB
class branchNbound:
    def __init__(self, puzzle = Puzzle()):
        self.puzzle = puzzle # simpan state awal puzzle
        self.goal = Puzzle() # simpan state goal
        self.goal.generate_goal() # mengenerate goal
        self.pq = PriorityQueue() # definisi awal priority queue
        self.leaf = None # definisi awal leaf (simpul yang diambil)
        self.table_node = {} # definisi awal hash tabel untuk menyimpan
node
        self.result = None # definisi awal result (simpul goal)
        self.time = 0 # definisi awal waktu untuk menyimpan waktu proses

    # algoritma BnB
    def solve(self):
        # check apakah puzzle dapat diselesaikan
        if (self.puzzle.sigma_kurang%2 != 0):
            # jika sigma kurang ganjil puzzle tidak dapat diselesaikan,
raise error
```

```
            raise Exception("\nCan't solve puzzle")
        print("Searching for solution...")
        # menyimpan waktu mulai proses
        start = time()
        # membangkitkan simpul akar
        initial = self.puzzle
        final = self.goal
        root_node = Node(None, initial, 0, self.calculateCost(initial,
final), [])
        # menambahkan simpul akar ke priority queue
        self.pq.enqueue(root_node)
        # insialisasi jumlah iterasi
        iterasi = 0
        while (not self.pq.empty()):
            if (iterasi < 1000000):
                # mengambil simpul pada queue paling awal
                minimumNode = self.pq.dequeue()
                # leaf adalah simpul yang sedang diproses saat ini
                self.leaf = minimumNode
                self.puzzle = self.leaf.puzzle
                # check apakah puzzle sudah mencapai goal
                if (self.is_final(self.puzzle)):
                    # menyimpan waktu akhir proses
                    end = time()
                    # jika mencapai goal prunning leaf yang memiliki cost
lebih besar cost saat ini
                    self.prunning(minimumNode.cost)
                    # simpul goal adalah simpul yang sedang diproses saat
ini
                    self.result = self.leaf
                    break
                else:
                    # mengecek proses pergeseran yang aman
                    swap_list = self.puzzle.safe_swap()
                    # simpan puzzle pada simpul yang sedang diproses
                    current_puzzle = self.puzzle
                    # mengambil semua swap yang aman
                    for move in swap_list:
                        if (move != '-'):
                            self.puzzle = deepcopy(current_puzzle)
                            # lakukan swap terhadap puzzle
                            self.puzzle.swap(move)
                            # tambahan informasi yang diperlukan node
seperti parent, depth, dan list_move
                            new_depth = minimumNode.depth + 1 # f(i)
                            new_cost = self.calculateCost(self.puzzle,
final) + new_depth # c(i) = f(i) + g(i)
```

```
                            new_move = minimumNode.list_move + [move]
                            new_node = Node(minimumNode, self.puzzle,
new_depth, new_cost, new_move)
                            # mengecek apakah simpul yang sedang diproses
sudah ada di hash tabel node
                            str_new_puzzle = self.puzzle.puzzle_to_string()
                            if (str_new_puzzle not in self.table_node):
                                # jika belum ada di hash tabel node, maka
tambahkan simpul tersebut ke hash tabel node dan priority queue
                                self.pq.enqueue(new_node)
                                self.table_node.update({str_new_puzzle:(new
_node.id)})
                    iterasi += 1
            else:
                # jika iterasi melebihi batas maksimal, raise error
                raise Exception("\nReach max iteration")

        # menyimpan semua informasi hasil yang diperlukan (untuk keperluan
GUI)
        self.time = end - start
        list_puzzle =  self.result.node_to_list()
        list_move = self.result.list_move
        total_vertice = len(self.table_node)
        total_move = self.result.depth
        return list_puzzle, total_move, list_move, total_vertice, self.time

    # menghitung taksiran cost g(i)
    def calculateCost(self, puzzle, goal):
        cost = 0
        for i in range(len(puzzle.buffer)):
            # jika ubin tidak sama dengan posisinya pada goal cost akan
ditambahkan
            if (puzzle.get(i) != goal.get(i) and puzzle.get(i) != 16):
                cost += 1
        return cost

    # melakukan pemotongan terhadap node yang memiliki cost lebih besar
dari cost saat ini
    def prunning(self, current_best):
        temp = PriorityQueue()
        for i in range(len(self.pq.elements)):
            if (self.pq.elements[i].cost <= current_best):
                temp.enqueue(self.pq.elements[i])
        self.pq = temp

    # mengecek apakah puzzle sudah mencapai goal
    def is_final(self, new_puzzle):
```

```
        # jika buffer puzzle sama dengan buffer goal maka puzzle sudah
mencapai goal
        return new_puzzle.buffer == self.goal.buffer

    # melakukan print hasil
    def print_result(self):
        print("\n\nCompletion Step : ")
        # print semua node lintasan dari akar ke simpul goal
        self.result.printall()
        print("Total move :", self.result.depth, "step")
        print("All moves : ", end="")
        for i, move in enumerate(self.result.list_move):
            print(move.capitalize(), end="")
            if (i != len(self.result.list_move) - 1):
                print(", ", end="")
        print()
        print("Total vertices raised : ", len(self.table_node))
        print("Completion time : %.4f second" %(self.time))
```

3.3.Puzzle15Solver.py

```
from lib import *
from bnb import *

if __name__ == '__main__':
    succ = False
    puzzle, prog = None, None
    print("######  #     # ####### #######
#       #######      #   #######")
    print("#     #
#     #      #       #  #       #           ##   #      ")
    print("#     # #     #     #       #   #       #          #
#   #      ")
    print("######  #     #    #       #    #       #####        #   ######
")
    print("#       #     #   #       #     #       #           #         #
")
    print("#       #     #  #       #      #       #           #   #     #
")
    print("#        #####  ####### ####### ####### #######    #####  #####
")
    print()
    print("        #####  ####### #       #     # #######
######             ")
    print("       #     # #     # #       #     #
#       #     #           ")
```

```
    print("       #       #     # #       #     #
#       #     #            ")
    print("        #####  #     # #       #     #
#####   ######             ")
    print("             # #     #
#        #   #  #       #   #              ")
    print("       #     # #     # #         #
#   #       #    #             ")
    print("        #####  ####### #######    #    #######
#     #            ")

    while True:
        print("============================================================
==========")
        # input pilihan menu
        menu = int(input("Choose Input Method\n1. Input Keyboard\n2. Input
File\n3. Generate Random\n4. Exit\n>> "))
        match menu:
            case 1: # input keyboard
                buffer = []
                print("Example input : ")
                print("3\t11\t4\t8")
                print("1\t2\t10\t16")
                print("6\t5\t7\t9")
                print("13\t14\t15\t12")
                print("Input Puzzle 15 : ")
                try:
                    c11, c12, c13, c14 = input().split()
                    c21, c22, c23, c24 = input().split()
                    c31, c32, c33, c34 = input().split()
                    c41, c42, c43, c44 = input().split()
                    succ = True
                except:
                    print("Error: Input not valid")
                buffer = [int(c11), int(c12), int(c13), int(c14),
                          int(c21), int(c22), int(c23), int(c24),
                          int(c31), int(c32), int(c33), int(c34),
                          int(c41), int(c42), int(c43), int(c44)]
                puzzle = Puzzle(buffer)
                for n in range(len(buffer)):
                    if buffer[n] == 16:
                        i, j = int(n/puzzle.col), n%puzzle.col
                        break
                puzzle.setX(i,j)
                puzzle.sigma_kurang = puzzle.kurang()
            case 2: # input file
                puzzle = Puzzle()
```

```
                try:
                    puzzle.read(input("Input file name (without .txt) : "))
                    succ = True
                except Exception as e:
                    print(e)
            case 3: # generate random puzzle
                puzzle = Puzzle()
                puzzle.generate_puzzle(randint(10,60))
                succ = True
            case 4: # keluar dari program
                exit()
            case _: # input menu tidak valid
                print("Error: Input not valid")
        # lakukan proses jika input sukses
        if (succ):
            succ = False
            puzzle.printInfo()
            prog = branchNbound(puzzle)
            try:
                prog.solve()
                prog.print_result()
            except Exception as e:
                print(e)
            puzzle, prog = None, None
        print("============================================================
==========\n")
```

3.4.Puzzle15Solver.pyw

```
from bnb import branchNbound
from lib import *
from PySimpleGUI import *
from os import getcwd

# membuat interface
def generate_interface():
    # mendapatkan directory saat ini untuk keperluan browse file
    cwd = getcwd()
    theme_background_color = '#8A360F'
    theme_text_color = '#F5F5F5'
    theme_button_color = "#CDAA7D"

    # membuat layout title
    title = [
        [
            Push(background_color=theme_background_color),
```

```
            Text('Puzzle Solver', font='Copperplate 35 bold',
text_color=theme_text_color, justification ="center",
background_color=theme_background_color),
            Push(background_color=theme_background_color),
        ],
        [
            HorizontalSeparator(),
        ]
    ]

    # membuat layout browse file
    browse_puzzle = [
        [
            Push(background_color=theme_background_color),
            # button generate random puzzle
            Button('Generate Puzzle', key="-GENERATE-",
size=(14,1),button_color=theme_button_color, font='Copperplate 9 bold'),
                        Input(size=(30,1), key="-BROWSE FILE-",
enable_events = True, tooltip="Browse for a text file"),
             # browse file
            FileBrowse(file_types=(("Text Files", "*.txt"),), key="-BROWSE-
", initial_folder=cwd ,size=(15,1) ,button_color=theme_button_color,
font='Copperplate 9 bold'),
             Push(background_color=theme_background_color),
        ],
    ]

    # membuat layout puzzle info
    puzzle_info = [
        [
            Text(key="-PUZZLE INFO-", font='Copperplate 11',
text_color=theme_text_color, background_color=theme_background_color),

        ],
        [
            Text(key="-X-", font='Copperplate 11',
text_color=theme_text_color, background_color=theme_background_color),
        ],
        [
            Text(key="-LIST KURANG-", font='Copperplate 11',
text_color=theme_text_color, background_color=theme_background_color),
        ],
        [
            Text(key="-KURANG-", font='Copperplate 11',
text_color=theme_text_color, background_color=theme_background_color),
        ],
        [
```

```
            Push(background_color=theme_background_color),
        ],
        [
            Text(key="-MOVE-", font='Copperplate 11',
text_color=theme_text_color, background_color=theme_background_color),
        ],
        [
            Text(key="-TOTAL MOVE-", font='Copperplate 11',
text_color=theme_text_color, background_color=theme_background_color),
        ],
        [
            Text(key="-VERTICES-", font='Copperplate 11',
text_color=theme_text_color, background_color=theme_background_color),
        ],
        [
            Text(key="-TIME-", font='Copperplate 11',
text_color=theme_text_color, background_color=theme_background_color),
            Push(background_color=theme_background_color),
            Button("New Game", key="-NEW GAME-", visible=False,
size=(12,1),button_color=theme_button_color, font='Copperplate 9 bold'),
        ],
    ]

    # membuat layout button proses
    button_proses = [
        [
            Push(background_color=theme_background_color),
        ],
        [
            Push(background_color=theme_background_color),
            # button mencari solusi (belum di visualisasi)
            Button("Find Solution", key="-FIND-", disabled=True,
size=(12,1),button_color=theme_button_color, font='Copperplate 9 bold'),
            # button untuk visualisasi puzzle
            Button("Visualize", key="-VISUALIZE-", disabled=True,
size=(12,1),button_color=theme_button_color, font='Copperplate 9 bold'),
            Push(background_color=theme_background_color),
        ],
        [
            Push(background_color=theme_background_color),
        ],
        [
            Push(background_color=theme_background_color),
            # slider untuk menentukan kecepatan animasi
            Text("Animation Speed ", font='Copperplate 11',
text_color=theme_text_color, background_color=theme_background_color),
```

```
            Slider(key="-SLIDER-", range=(1500,100), default_value=250
,orientation="h", size=(15,7), disable_number_display=True, disabled=True,
background_color=theme_button_color),
            Push(background_color=theme_background_color),

        ],
    ]

    Puzzle_Coloum_1 = [
        [Image(key = "-IMAGE1-", filename="./assets/16.png"
,background_color=theme_background_color)],
        [Push(background_color=theme_background_color),],
        [Image(key = "-IMAGE5-", filename="./assets/16.png",
background_color=theme_background_color)],
        [Push(background_color=theme_background_color),],
        [Image(key = "-IMAGE9-", filename="./assets/16.png",
background_color=theme_background_color)],
        [Push(background_color=theme_background_color),],
        [Image(key = "-IMAGE13-", filename="./assets/16.png",
background_color=theme_background_color)],
    ]

    Puzzle_Coloum_2 = [
        [Image(key = "-IMAGE2-", filename="./assets/16.png"
,background_color=theme_background_color)],
        [Push(background_color=theme_background_color),],
        [Image(key = "-IMAGE6-", filename="./assets/16.png",
background_color=theme_background_color)],
        [Push(background_color=theme_background_color),],
        [Image(key = "-IMAGE10-", filename="./assets/16.png",
background_color=theme_background_color)],
        [Push(background_color=theme_background_color),],
        [Image(key = "-IMAGE14-", filename="./assets/16.png",
background_color=theme_background_color)],
    ]

    Puzzle_Coloum_3 = [
        [Image(key = "-IMAGE3-", filename="./assets/16.png"
,background_color=theme_background_color)],
        [Push(background_color=theme_background_color),],
        [Image(key = "-IMAGE7-", filename="./assets/16.png",
background_color=theme_background_color)],
        [Push(background_color=theme_background_color),],
        [Image(key = "-IMAGE11-", filename="./assets/16.png",
background_color=theme_background_color)],
        [Push(background_color=theme_background_color),],
```

```
        [Image(key = "-IMAGE15-", filename="./assets/16.png",
background_color=theme_background_color)],
    ]

    Puzzle_Coloum_4 = [
        [Image(key = "-IMAGE4-", filename="./assets/16.png"
,background_color=theme_background_color)],
        [Push(background_color=theme_background_color),],
        [Image(key = "-IMAGE8-", filename="./assets/16.png",
background_color=theme_background_color)],
        [Push(background_color=theme_background_color),],
        [Image(key = "-IMAGE12-", filename="./assets/16.png",
background_color=theme_background_color)],
        [Push(background_color=theme_background_color),],
        [Image(key = "-IMAGE16-", filename="./assets/16.png",
background_color=theme_background_color)],
    ]

    # membuat layout puzzle board
    puzzle_board = [
        [
            Push(background_color=theme_background_color),
            Column(Puzzle_Coloum_1,background_color=theme_background_color)
,
            Column(Puzzle_Coloum_2,background_color=theme_background_color)
,
            Column(Puzzle_Coloum_3,background_color=theme_background_color)
,
            Column(Puzzle_Coloum_4,background_color=theme_background_color)
,
            Push(background_color=theme_background_color),
        ],
    ]

    # menyatukan semua layout
    Layout = [
        [
            title,
            browse_puzzle,
            puzzle_board,
            button_proses,
            puzzle_info,
        ],
    ]
    window = Window('Puzzle 15', Layout,
background_color=theme_background_color, size=(600,780))
    return window
```

```
if __name__ == '__main__':
    window = generate_interface()
    puzzle_board = None
    prog = None
    list_puzzle, total_move, list_move, total_vertice, time,
str_list_kurang = [], None, [], None, None, ""
    while True:
        event, value = window.read()
        if event == "EXIT" or event == WIN_CLOSED: # jika keluar dari
program
            break
        if event == "-BROWSE FILE-" or event == "-GENERATE-": # jika
memilih file puzzle atau mengenerate random puzzle
            try:
                puzzle_board = Puzzle()
                # cek apakah memilih file atau mengenerate puzzle
                if (event == "-GENERATE-"):
                    # membersihkan input browse file
                    window["-BROWSE FILE-"].update("")
                    # generate puzzle
                    puzzle_board.generate_puzzle(randint(10,60))
                else:
                    # membaca file puzzle
                    puzzle_board.read(value["-BROWSE FILE-"], True)

                row = puzzle_board.row
                col = puzzle_board.col

                # visualisasi puzzle awal dari hasil input
                for i in range (row*col):
                    window["-IMAGE"+str(i+1)+"-
"].update(filename="./assets/"+str(puzzle_board.buffer[i])+".png")

                # visualisasi informasi awal puzzle, yaitu nilai X, nilai
masing-masing kurang(i), dan sigma kurang(i) + X
                window["-PUZZLE INFO-"].update(" Puzzle Info : ")

                window["-X-"].update(" X : " + str(puzzle_board.X))

                str_list_kurang += " List Kurang : \n"
                for i in range(puzzle_board.row):
                    for j in range(puzzle_board.col):
                        str_list_kurang += str(" Kurang(%.2d) : "
%((i*Puzzle.col+j)+1)) + str("%.2d"
%puzzle_board.list_kurang[i*Puzzle.col+j]) + " "
                    if i != puzzle_board.row-1:
```

```
                        str_list_kurang += "\n"

                window["-LIST KURANG-"].update(str_list_kurang)
                str_list_kurang = ""

                window["-KURANG-"].update(" Sigma(Kurang(i)) + X : " +
str(puzzle_board.sigma_kurang))
                window["-FIND-"].update(disabled=False) # tombol find bisa
digunakan
            except Exception as e:
                window["-FIND-"].update(disabled=True)

        if event == "-FIND-": # jika menekan tombol find
            try:
                # proses pencarian
                prog = branchNbound(puzzle_board)
                list_puzzle, total_move, list_move, total_vertice, time =
prog.solve()
                list_puzzle.reverse()
                window["-VISUALIZE-"].update(disabled=False) # tombol
visualisasi bisa digunakan
                window["-SLIDER-"].update(disabled=False) # slide bisa
digunakan
            except Exception as e:
                # menampilkan eror yang di raise (iterasi mencapai max atau
tidak ada solusi)
                popup(e)

        if event == "-VISUALIZE-": # jika menekan tombol visualize
            animation_speed = value["-SLIDER-"]
            str_move = ""

            # proses visualisasi
            for i, puzzle in enumerate(list_puzzle):
                if (i != len(list_puzzle)-1):
                    # menampilkan move yang diambil
                    window["-MOVE-"].update(" Taken move : " +
str(list_move[i].capitalize()))
                for j, element in enumerate(puzzle.buffer):
                    window["-IMAGE"+str(j+1)+"-
"].update(filename="./assets/"+str(element)+".png")
                window.read(timeout=animation_speed)

            # menampilkan informasi puzzle setelah di visualisasi, yaitu
total move, total vertice, dan waktu yang dibutuhkan
            window["-TOTAL MOVE-"].update(" Total move : " +
str(total_move))
```

```
            window["-VERTICES-"].update(" Total vertice : " +
str(total_vertice))
            window["-TIME-"].update(" Time : %.4f" %time)
            window["-BROWSE-"].update(disabled=True) # tombol find di
disable
            window["-GENERATE-"].update(disabled=True) # tombol find di
disable
            window["-FIND-"].update(disabled=True) # tombol find di disable
            window["-NEW GAME-"].update(visible=True) # tombol new game
muncul

        if event == "-NEW GAME-": # menekan tombol new game
            # mengembalikan ke kondisi awal program
            for i in range(1,16):
                window["-IMAGE"+str(i)+"-
"].update(filename="./assets/16.png")
            window["-BROWSE FILE-"].update("")
            window["-PUZZLE INFO-"].update("")
            window["-X-"].update("")
            window["-LIST KURANG-"].update("")
            window["-KURANG-"].update("")
            window["-MOVE-"].update("")
            window["-TOTAL MOVE-"].update("")
            window["-VERTICES-"].update("")
            window["-TIME-"].update("")
            window["-GENERATE-"].update(disabled=False)
            window["-BROWSE-"].update(disabled=False)
            window["-NEW GAME-"].update(visible=False)
            window["-VISUALIZE-"].update(disabled=True)
            puzzle_board = None
            prog = None
            list_puzzle, total_move, list_move, total_vertice, time = [],
None, [], None, None
    window.close()
```

Lampiran

1. Alamat Kode Program (*Github*)

https://github.com/Stanley77-web/Tucil3_13520028.git

2. Tabel Checklist

Poin	Ya	Tidak
1. Program berhasil dikompilasi	√	
2. Program berhasil running	√	
3. Program dapat menerima input dan menuliskan output.	√	
4. Luaran sudah benar untuk semua data uji	√	
5. Bonus dibuat	√	

3. Contoh Instansiasi Puzzle

a. tc-1.txt

```
1 6 3 4
9 16 2 7
10 5 11 15
13 14 12 8
```

b. tc-2.txt

```
6 2 7 3
1 5 11 4
9 10 16 8
13 14 15 12
```

c. tc-3.txt

```
3 6 2 4
1 9 8 11
10 5 15 16
13 7 14 12
```

d. tc-4.txt

```
5 2 8 10
1 11 6 4
7 9 16 3
13 14 15 12
```

e. tc-5.txt

```
7 1 3 16
9 2 12 4
6 5 14 11
13 15 10 8
```